UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Vladislav Stafinjak

Expanding the functionalities of VREX

Bachelor's Thesis (9 ECTS)

Supervisors:
Jaan Aru
Madis Vasser
Raul Vicente Zafra

Tartu 2016

Expanding the functionalities of VREX

Abstract:

Conducting psychological experiments in the real world has limitations like cost, real world physics, lack of control and many more. Current technologies give us all the needed tools to create virtual environments that are free of these limitations and VREX is a toolbox that tries to use these advantages of technology. During this thesis VREX was reworked and made into a toolbox that is distributed as a Unity project which has all the essential blocks to create new experiments. Functionalities like new locomotion systems and the possibility to use audio files in experiments have been added. Editor view has been reworked and improved and code has been commented to ease the creation of new custom experiments. In addition to well explained code a blank experiment was created that has all the essential building blocks that any experiment needs so that users can effortlessly build their own experiments.

Keywords:

Virtual reality, Unity, psychology, toolbox

CERCS: P175

Tööriistakasti VREX edasi arendus

Lühikokkuvõte:

Psühholoogiliste eksperimentide läbiviimine päris maailmas on piiratud hinna, füüsikaseaduste, kontrolli vähesuse jms tõttu. Tänapäeva tehnoloogia pakub vajalikke võimalusi, et luua keskkondi virtuaalses reaalsuses, mis on vaba nendest piirajatest. VREX on tööristakast, mis üritab kasutada uusi tehnoloogiaid. Käesoleva töö käigus muudeti algul eraldiseisva programmina olnud VREX Unity tööriistakastiks, millel on kaasas kõik vajalikud klotsid, et luua uusi psühholoogilisi eksperimente. VREX'i lisati mitu uut vajalikku funktsionaalsust. Lisati meetodeid katseisiku liikumiseks, audio failide tugi, muudeti 3D-redigeerimise aken ja dokumenteeriti kood paremini, et uute eksperimentide loomine oleks lihtsam. Lisaks funktsionaalsustele loodi ka tühi näidiseksperiment, milles on kõik vajalik eksperimendi loomiseks juba olemas. Seda tühja näidist kasutades on kasutajal lihtne VREXi abil uut eksperimenti luua.

Võtmesõnad:

Virtuaalne reaalsus, Unity, psühholoogia, tööriista kast

CERCS: P175

Table of Contents

1 Introduction

Experimental psychology studies our cognition, but most of the studies have been using very simple stimuli and have been done with subjects sitting in front of the computer monitor. It is unclear how much such experiments can tell about real-life cognition. Conducting more ecologically valid psychological experiments has been very resource-hungry during the last decades. It takes a lot of effort and time to create the needed environments, especially when the study is to be conducted in real-life environments. Furthermore, in such real-world settings gathering enough participants is cumbersome, as the environments are difficult to move. In addition, due to experiments often being very different every experiment will most likely need the creation of custom environment.

Psychological experiments often need to be conducted in closed systems so that no outside confounding factors can affect the outcome of the study. But in the real world it is hard to control all the interferences and often experiments need to be repeated due to some outside factor corrupting the data.

Advancements in technology and computer science now make it possible to create immersive quasi-realistic environments in virtual reality that are well-controllable and hence suitable for experimental studies. Using virtual reality will give the researcher complete control of the environment and allow one to create situations that cannot be conducted in the real world.

It has been shown that virtual reality creates enough sense of immersion that it can capture many aspects of our cognition [1]. Thus, virtual reality is a viable way to conduct scientific research in the field of psychology.

However, starting from scratch and building everything by oneself can be difficult and also time consuming. This will involve knowledge of different programming languages and platforms like Unity or Unreal Engine, time to learn different API's and read their documentations. Finding a programmer that can do all this alone in reasonable time can be difficult or costly.

This is where VREX comes in. VREX is a toolbox that was created to ease the creation of psychological experiments that use virtual reality environments. It is built on Unity and distributed as a project not a standalone application. VREX gives users a way to create virtual reality environments and with little bit of coding create custom experiments that can be tailored to work exactly like needed. VREX provides environment creation and automatic generation to create similar but still different environments. VREX also provides users with pre-built functionalities with reasonable documentation that can be accessed using doxygen plugin for Unity and comments in code that will ease the creation of new unique functionalities if needed. An average computer science student and, more importantly, an average experimental psychologist should handle the creation of new experiments using the examples and functionalities that are provided by VREX in a reasonable time.

In this thesis we create a new set of capabilities to improve the design of experiments and the user-experience in VREX to the level of allowing scientists to conduct rigorous psychological experiments. In particular, we developed new modules of locomotion, reworked 3D environment editor view movement system, added tools to use a room based audio and created a empty template for new experiments.

This thesis will be divided into three main paragraphs "Background and related work", "The toolbox" and "General discussion". The paragraph "Background and related work" talks about the background of used technologies and virtual reality itself. "The toolbox"

concentrates on work done during this thesis. And the last paragraph “General discussion” will discuss about the current state of the VREX and what should be done in the future.

2 Background and Related Work

2.1 Virtual Reality

Virtual reality is an artificial environment that is created using software and presented to the user in a way that user starts to believe and accept the environment as a real environment. Most of the times virtual reality is experienced through two of the five senses: sound and sight. More rarely the vestibular system (sensory mechanism in our inner-ear that detects the movement and tilt of our head and thus helps our body with balance) is used which creates a more realistic experience for the user thus increasing the immersion level of the experience. In the last few years new technological advances have been emerging that offer greater immersion and presence that traditional virtual reality technology has given so far.

And the increase of popularity of virtual reality technology can be easily shown by the numbers that Oculus got during their initial Kickstarter launch. Within 24 hours, they raised $670 000 from 2750 people and soon broke the 1 million barrier [2]. Oculus being the resurrector of an entire genre got a headstart and sold around 175 000 head mounted devices by the year 2015 [3].

2.2 Oculus Rift

The Oculus Rift is a virtual reality headset or HMD (head mounted device) developed and manufactured by Oculus VR. It was released on 28 March 2016, making it the first to kickstart consumer-targeted virtual reality headsets.

This paragraph will compare different versions of oculus rift and list system requirements given by Oculus. But further users of VREX need to keep in mind that this toolbox will have lower requirements if used with smaller and more optimized 3D objects.

2.2.1 Development Kit I vs Development Kit II

As you can see on Table 1 Oculus Rift DK2 has improved resolution and refresh rate. We recommend using at least DK2 because we ourselves did not have access to DK1 to test the compatibility with VREX. The requirements for development kits were took from their own website [4-5].

Table 1. *Oculus rift development kits specifications.*

	Oculus rift DK1	Oculus rift DK2
Resolution	640 x 800 per eye	960 x 1080 per eye
Refresh Rate	60 Hz	72 Hz, 60 Hz
Persistence	~3ms	2ms, 3ms
Viewing Optics	110° Field of View (nominal)	100° Field of View (nominal)
Positional tracking	CMOS Sensor	Near Infrared CMOS Sensor

2.3 System requirements for VREX

As VREX needs a lot less resources that Oculus itself needs to work on a given PC, we based our own requirements on the most popular virtual reality head mounted devices Oculus Rift and HTC Vive to list approximate requirements.

Oculus Rift recommended [6]:

- NVIDIA GTX 970 / AMD 290 equivalent or greater
- Intel i5-4590 equivalent or greater
- 8GB+ RAM
- Compatible HDMI 1.3 video output
- 2x USB 3.0 ports
- Windows 7 SP1 or newer

HTC Vive recommended [7]:

- NVIDIA GeForce® GTX 970 / AMD Radeon™ R9 290 equivalent or greater
- Intel i5-4590 / AMD FX 8350 equivalent or greater
- 4GB+
- HDMI 1.4 or DisplayPort 1.2 or newer
- 1x USB 2.0 or greater port
- Windows 7 SP1 or newer

Based on their combined requirements is is safe to say that VREX can be used on a system that meets the following requirements.

Combined requirements :

- NVIDIA GTX 970 / AMD 290 equivalent or greater
- Intel i5-4590 / AMD FX 8350 equivalent or greater
- 8GB+ RAM
- 2x USB 3.0 ports
- Windows 7 SP1 or newer

However, as VREX actually uses a lot less resources, it can most likely be used on any computer that can run Oculus or HTC Vive.

2.4 Prior experiments using virtual reality

Next I will explain some selected findings that demonstrate the usefulness of virtual reality for studying cognition.

Schultheis and others [8] found in their research that virtual reality creates enough feeling of immersion that it can capture many aspects of our cognition. This shows that virtual reality creates enough of sense of reality in participants and thus virtual reality could be used as a tool to conduct psychological experiments. Another finding that shows the potential of virtual reality was presented by Hoffman and others [9]. In their research it was found that using virtual reality lowered the discomfort and pain in participants. Thus virtual reality can be a feasible way of managing and controlling adjunctive pain. They also mentioned that all this was done using ~400 USD head mounted device Oculus and they were very pleased by the cost-effectiveness of such solutions.

Felnhofer and others [10] found that in contrast to traditional mood induction procedures, virtual environments provide us with richer and more prevalent stimuli therefore virtual environments can elicit specific emotions more easily and efficiently. From this we can conclude that virtual reality is a better solution that other 2D solutions.

As virtual reality is getting more and more popular there is an increasing number of studies being done.

3 The toolbox

This chapter will briefly talk about the state of the VREX prior to this thesis and then list all the new functionalities and other work that was done during this thesis.

3.1 State of VREX prior to this thesis

It was created on Unity 4 and a clean version of their project was handed to us over their git repository. VREX had two working psychological experiment types: change blindness and memory. It was one of the goals to preserve the functionalities of these experiments.

After a small fix the project was carried to Unity 5 and saved as a reference point for further comparison [Appendix 1].

The biggest problem that we encountered was that even though the experiments themselves were written reasonably well the code was not well commented so we needed to read through the code to understand the issues. The not-so-optimal documentation is understandable as the previous developers were still developing VREX with an idea that this will be a standalone application.

3.2 Work done during this thesis

This chapter will list things that were created and developed only by the author of this thesis. Functionalities that were added were all discussed and chosen in collaboration with the whole team. All added new objects or scripts that are referred in this paragraph can be found in the VREX project [Appendix 2] simply by opening the Unity project and under "Project" tab using Assets search bar to find objects by name.

It should be kept in mind that this thesis will only explain the added functionalities and describe how they work and what they do from the front-end. Description about how the code itself works is added to the code files using comments that are picked up by documentation generators.

Transferring to Unity3D 5

Transferring to Unity 5 was a easy step but a needed one. Unity 5 is a free software that has Personal Edition. This gives us a chance to promote and distribute VREX not as a standalone software but as a Unity 5 project. This way we as developers have more freedom and future users will not be restricted to only those functionalities that we have implemented but have a chance to implement their own ideas and further develop VREX.

Unity 5 has a greater support for virtual reality and thus a better version to develop our toolkit for.

Locomotion system

The locomotion system that was used in the previous VREX version caused severe nausea in some of the testers. Therefore, developing a new locomotion system that would not cause discomfort was needed. The old locomotion system was causing nausea because of many different aspects. One of the main reasons was that both the movement of the headset and the mouse moved the field of view which caused conflicts between visual stimuli and bodily senses.

Developers of Oculus have published an article talking about simulator sickness [11]. They brought out 10 main factors that can cause simulator sickness.

They are :

- Acceleration
    - Acceleration can cause conflicts between seeing the movement but not feeling it with other senses
    - In our case so far irrelevant because we are dealing with fixed speeds and small environments
- Degree of control
    - Can cause the user to feel not in control and thus give body signals that something is wrong.
- Duration of simulator use
    - Irrelevant in our case.
- Altitude
    - Avoid filling the field of view with the ground.
- Binocular disparity
    - Some find viewing stereoscopic images uncomfortable
- Field-of-View
    - Reducing the amount of visual field lower than humans normal field of view may cause conflicts between sensory input and expectations.
- Latency
    - Dropping frame rate and lags can cause conflicts between the input and what our brain is used to.
- Distortion correction: use Oculus VR's distortion shaders
    - VREX is using OVR controllers that are made specially for VR experience.
- Flicker: do not display flashing images or fine repeating textures
    - This can be countered just by keeping this in mind when designing environments.
- Experience: experience with VR makes you resistant to simulator sickness (which makes developers the worst test subjects)
    - Should be considered by developers when testing their solutions.

We looked at many different locomotion systems that are being used in different virtual reality demos or games. Based fully on subjective thoughts locomotion system called "Blink" developed by Cloudhead Games [12] was chosen as a base or an example for our own locomotion system. Based on that locomotion system a new locomotion system was made that suits our needs [Appendix 3].

Our locomotion system can be configured in 4 different ways:

- Smooth movement and turning (SMST)
- Smooth turning but incremented movement (STIM)
- Smooth movement but incremented turning (SMIT)
- Incremented movement and turning (IMIT)

To find out which of these configurations should be used in VREX a small questionnaire was done.

Questionnaire Setup

A special environment was created inside Unity 5 engine to test four different locomotion systems. Environment consisted of system of rooms with some hidden objects that user had to find. The room system can be seen on Figure 1.

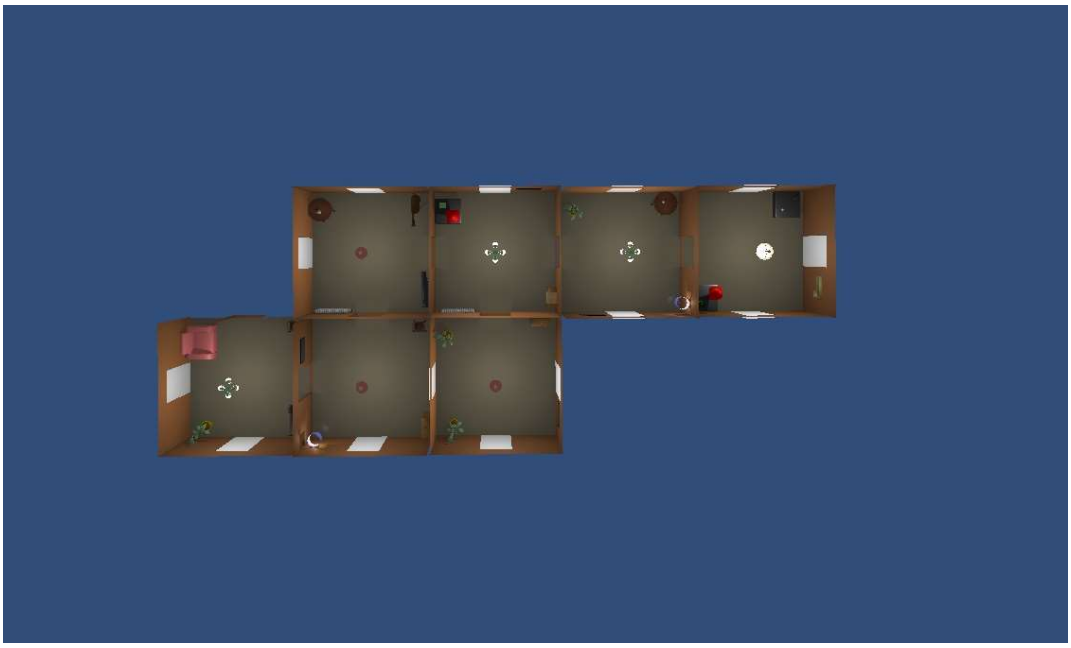

Figure 1. *Environment (view from above) for testing different locomotion systems.*

User tried out all four different locomotion options for about 5 minutes eache and after every one of them they were asked to fill in a questionnaire. The small experiment was conducted with 10 subjects and the results are shown in Table 2.

After every test the subject had to answer these questions:

Did you feel any nausea?

Did you feel any discomfort?

Comments.

After all four test are conducted the subject had to choose the best locomotion system.

Results of the locomotion questionnaire

Table 2. *Results from locomotion questionnaire*

	SMST	STIM	SMIT	IMIT
Nausea (number of participants that experienced nausea)	4	1	2	1
Discomfort (number of participants that experienced discomfort)	6	4	2	4
Number of users that chose that solution	2	2	4	2

There were some recurring comments about incremented movement. It felt unnatural and some participants even experienced some discomfort (40%).

The worst result was caused by the SMST locomotion system as it caused nausea in 40% of the participants and overall discomfort in 60% of the participants.

Our goal was to find which locomotion system would not cause any nausea or discomfort. But this small questionnaire showed that different persons react differently to these locomotion systems and thus it is best to add all four different options that can be fully customizable by the end-user.

Our final solution

We used the default OVRPlayerController that is made for virtual reality applications by Unity [13]. OVRPlayerController has many functionalities that we needed already implemented in them. My work was to add newly needed functionalities and integrate previously made player controllers so that all the previous functionalities that used the old solution now work with the new one. And further changes were made when new functionalities were added that needed some player controller tweaking (e.g. predetermined movement, see below 3.2.5).

Added functionalities

Smooth movement is already working on OVRPlayerController and so is the incremented rotation. A toggle button in OVRPlayerController Inspector was added called "*Incremented Movement*" that turns on the incremented movement and with that two fields that control the speed and the step size of the incremented movement in addition for testing purposes a toggle was added to toggle mouse movement on Y axis called "*Vertical Mouse Movement*"

that should be always disabled when using VR HMD's. Setting for OVRPlayerController can be seen on a Figure 3.

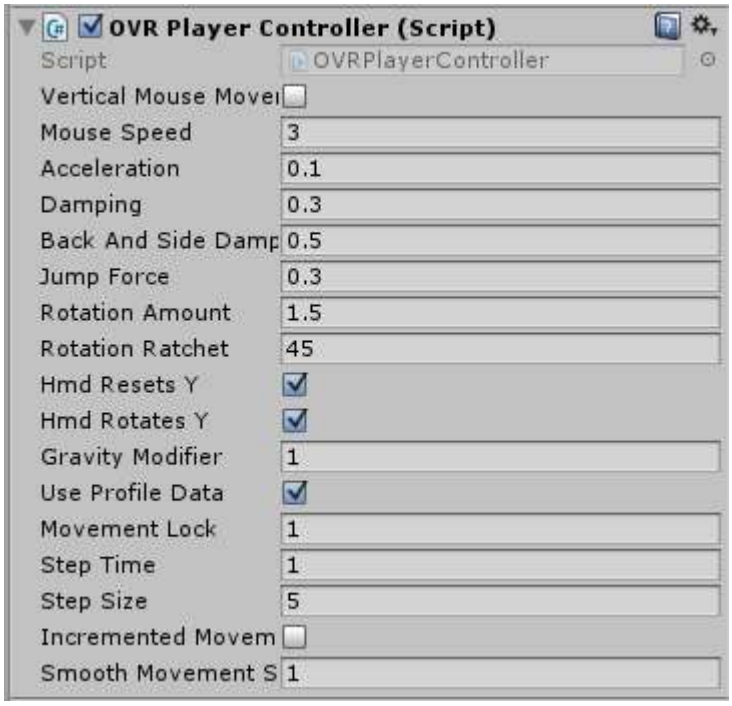

Figure 3. *A screenshot of Unity inspector view for OVR Player Controller modified during this thesis. First toggle „Vertical Mouse Move..“ allows to use mouse-look on vertical axis. Next addition is „Movement Lock“: turning this to „0“ will stop player from moving. Incremented movement is toggled on and off using „Incremented Movem…“ toggle and then customised using „Step Time“ and „Step Size“. „Step Time“ is a variable that says how long the pause should be between two steps. „Step Size“ sets the distance that can be covered in one step. „Smooth Movement S…“ is a variable that manages the speed of movement while incremented movement is turned off. This should be between 1 and 1.2*

Previously used cameras that were tracking where player watches to allow selecting objects in view were integrated into OVRPlayerController so that all the functionalities stayed working. In addition another player controller was created that did not have change blindness scripts attached so that it could be used outside of change blindness experiment. This was needed because change blindness experiment used its own custom scripts on player controller and using them outside of that change blindness scene would cause those custom scripts to throw errors. This second player controller should be used in new experiments.

Reworked 3D editor view movement

3D editor view allows users to modify already made environments and add new items into them. 3D editor is needed to add custom objects that have to be added manually (e.g. room based audio, see below 3.2.4).

Old solution

Old solution was using 'W','A','S','D' keys to control movement of objects and mouse drag up and down moved the camera view.

Cons:

- Unintuitive
- Lacking in precision of control

Our solution

We tried to make the 3D editor view movement as simple as possible.

Interactions in 3D edit view are divided into two subcategories:

- While left mouse button held down:
  - User can move using W,A,S,D keys or arrow keys
  - Manipulate his speed with Left Shift and Left Ctrl
    - Left Shift - raising the speed
    - Left Ctrl - lowering the speed
  - Look around using mouse movement
- While left mouse button is not held down:
  - Left click on the object selects the object
  - Right click deselects any selected objects.
  - W,A,S,D keys and arrow keys can be used to move, rotate or change size of the selected object, depending on the selected functionality in Modification Panel
  - Keys Q and E will change object's position on a vertical axis.

Our solution was based only on my own experience and later tested with other participants of the development team. It seemed as a easy and learning curve free solution. The idea itself was taken from previously experienced camera view systems from other applications - mostly typical first-person shooter game free camera modes and our solution is also similar to Unreal Engine's control scheme.

Figure 4. *On this screenshot AudioSphere is located in upper right quarter and is colored green. The white lines show the field of view of AudioSphere. If an object with „player“ tag will enter AudioSphere's field of view AudioSphere will launch its audio clip.*

Added room based sound

One of my tasks was to add sound that is room based. Our goal was to make it so that when creating an environment user can choose a certain audio file played in certain locations. For that an object called AudioSphere was made. It is a addable object that can only be added manually. Room furniture generator will not automatically generate it. User needs to open the environment in 3D edit view and add the object by hand.

AudioSphere works by reading in all "*.ogg" files from "*./Resources/AudioFiles*" directory and presenting them to user in a way of dropdown menu in 3D edit view. In 3D edit view user can listen to the audio files and select the needed one. Later during runtime (when an experiment is launched) AudioSphere will look for an object with "player" tag and if he finds one and he is in given range (by default it is 5 Unity units) an audio source will be given a selected audio clip and launched.AudioSphere will only look for "player" tagged objects that he can see as shown on Figure 4.

This audio source will play until the audio clip ends. All the inner working of AudioSphere are in script called "*Music Script.cs*".

The reasoning behind room base system is to provide player with audio guides that can be turned

on based on the room where user is. And this gives users a way to implement room based sound effects that can trigger different actions or emotions.

Pre-determined movement

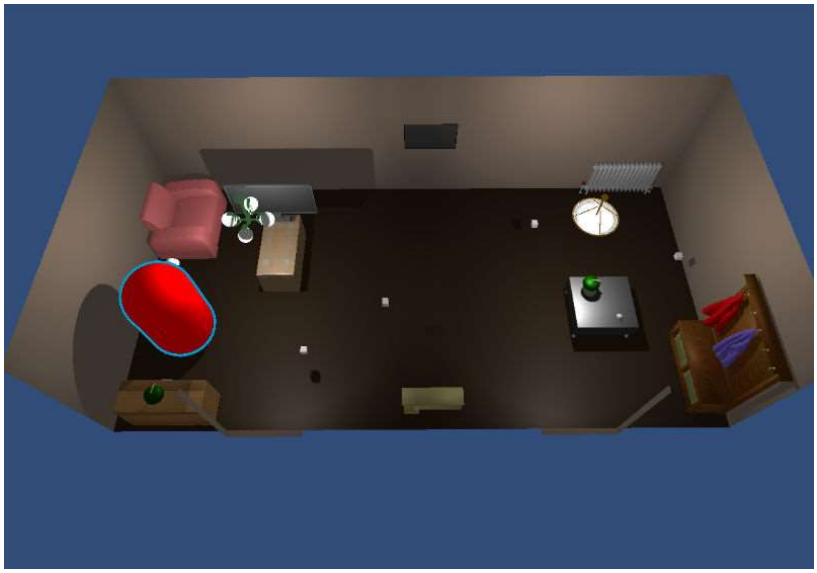

Figure 5. *The red capsule is the starting point of the Player. Predetermined movement is controlled using white boxes seen on the screen. They are freely movable objects. At runtime player will start slowly moving towards the closest white box and upon reaching certain distance destroys the closest square.*

Predetermined path was needed in case user wants to limit players time in each room by fully taking this under the experimenter's power. To allow moving player during experiments on a given path a *"movementPointer"* prefab was created. It is a small game object that can be added in environment 3D edit view. User can create a path of multiple

"movementPointers" that can be visible only in 3D edit view as shown on a Figure 5. *"Hide If Player"* script is used to hide it during experiment.

Movement itself is done by a script called *"movePlayer"* which is connected to a OVRPlayerController called "*OurMovementController.cs*" that was modified for our needs.

This script will look for "*moveSpot*" tag on a given scene and if found will lock player movement but keep its rotation ability so that the player can still look around. Then the script will calculate the closest pointer to a player and start moving player to the closest one at a steady speed. When the player gets into a given range of a pointer the pointer is destroyed and a new one is found. When all the pointers are destroyed the player's movement is restored.

It was done using multiple objects because using spline methods it would be hard to implement into the runtime 3D editor and step by step objects will allow later users to customize them so that the experimentator can choose when the player should stop or move on by just adding a on click trigger to every object.

For example, when the player gets close to pointer then it is not destroyed but player will just be stopped there until the experimentator presses a given key that will destroy that pointer and make the player move to the next one. Alternatively the user can customize it so that it will be time based - before destroying the pointer wait for 5 seconds. All that can be later added with only few lines of code.

Teleportation locomotive system

This locomotion system was created in later stages of development as an alternative locomotion system. Teleporting completely removes movement and thus lowers the chance on nausea or discomfort.

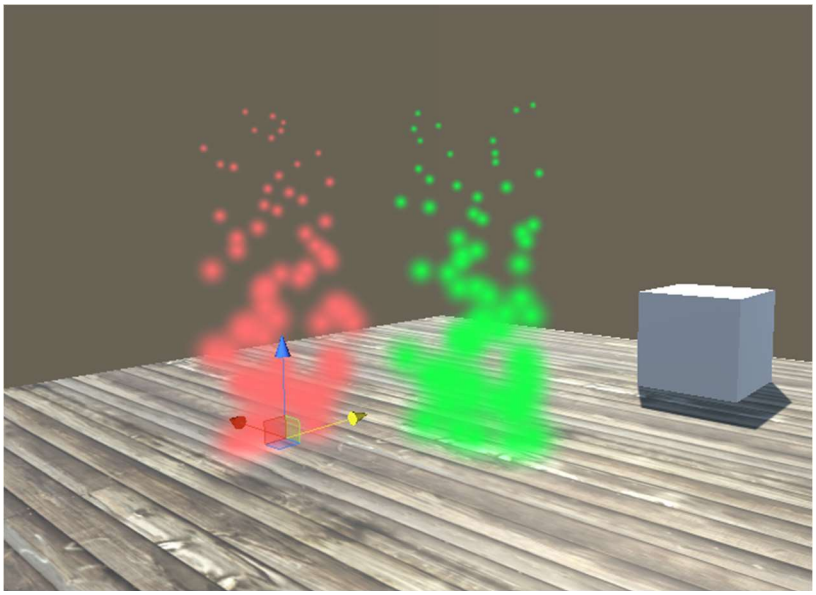

Figure 6. *Shows two different partice effects that are used in teleportation locomotion system. Green indicates that the chosen location has no obstacles and player can teleport there. On the other hand, red shows that teleporting there will cause collisions between player and other game objects thus teleporting there is not allowed.*

It is a example script that works by changing player's position on a click to where player is watching but before that it detects all collisions that can occur and if they occur player model is not teleported. For ease of use teleportation requires two presses of predetermined keys. First press will activate the script and show user a visual indication where he can be teleported my showing a marker that is colored green if the player is allowed to teleport there and red if the player is not allowed to teleport to given destination. Markers shown on a corresponding Figure 6.

This locomotion system is written in "*telePortationScript.cs*" script. This script can be as a locomotion system that needs only one button to function and by doing this it can remove the need of a keyboard for user. For example giving the player only the mouse to use will give player three buttons. One of them will be used for movement and two for interactions with the environment.

In addition to locomotive teleportation a way to change player's position using certain checkpoints was added that would give experimentators more tools to play with player's position. Script is called "*teleporterEnter.cs*" that uses two custom prefabs as checkpoints that can be added during 3D edit view. One acts as a teleport entry point and another as exit point. The only limitation is that only one pair of them can be used. Adding more teleport points can cause the script to not work as intended. This will allow to move player that enters the entry point to any other place in that environment.

Adding support for new experiments

Before this functionality was added only two types of experiments were allowed in VREX - memory experiment and change blindness experiment. However, ideally VREX should be a toolbox for others to make new experiments that might go beyond memory and change blindness experiments. To accomplish end result many smaller tweaks were made in the UI as seen on the Figure 7.

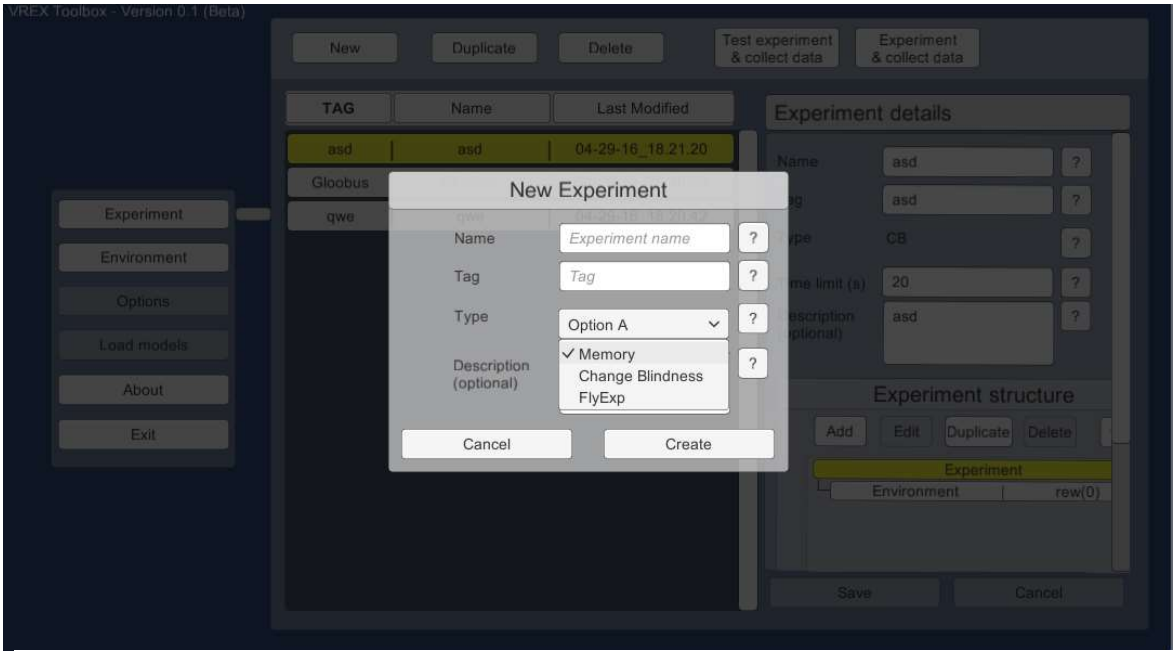

Figure 7. *This is a new experiment creation window. In earlier stages there were 2 toggles nest to „Type" but during this thesis it was reworked so that there is a dropdown box to allow more custom experiments.*

In the backend of this problem, scripts were altered ("*UIExperiment.cs*", "*UI Main Menu.cs*") to allow a dropdown solution for choosing a experiment type on creation.

It seemed as a simple functionality to add but a lot more work was needed mainly because the "*UIExperiment.cs*" and "*UI Main Menu.cs*" scripts were compiled from two different solutions as it seemed. And it was hard to keep all the functionalities working while adding support for multiple experiments. Reworking all that code was too time consuming so late in development so all the places in code that need to be changed when adding a new experiment are commented with tags and examples.

Adding new models into VREX

It can be that some of the experiments need new custom objects added into the environment. Or maybe the user wants more diversity in their environments. Thus, it was desired to add a way for users to add these new models.

At first we tried to make a runtime solution to add new objects into VREX. But only a ".obj" model importer was found to be free. We tried to add a solution to import new models with this importer called Runtime OBJ Importer by aaro4130 from Unity community [14].

We soon discovered that it was a slow solution and not very reliable because some of the tested models showed missing parts or materials and some crashed the VREX and Unity.

So far all the models were kept as prefabs inside Unity but we could not find any way to save new models as prefabs because prefabs can be created only in Unity editor and during runtime it is impossible to create them. Due to this it was considered that new models should be created inside Unity as prefabs or a complete overwork of "*SaveLevel.cs*" and "*LoadLevel.cs*" scripts would have been needed to be done.

After some more research looking for other possible solutions or other model formats that would be faster when loading it was decided that we will offer user a guide on how to add new models in Unity and integrate them into VREX. For that a scene called "*AddingObjects.unity*" was created and a "Guide" folder with guiding videos and readme text file [Appendix 4].

If in later stages of VREX it will be needed to add a way to import new models during runtime an "*modelImportingExample.cs*" script was created in "./Assets/Scripts/Examples" directory with some suggestions on how this can be accomplished if better importing methods will be found.

A clean template for new experiments

As the main goal of VREX is to allow users to create custom experiments. A clean and commented scene was created with all the essential building blocks needed for any of the experiments that could be done on VREX. The scene is called "*NewEXP.unity*".

The template loads environments and positions the player in the right place. All the items that are environment based (AudioSphere and predetermined movement) are still working.

In the code of the template all the more complex functions were commented so that it would be easier to understand for new users.

Experiment logging

As every experiment will need its own custom logging scripts it was decided that only basic logging will be added into VREX. They should act as examples for creation of new logging scripts or be used to get the most basic information about the experiments.

A "*saveSeenGO.cs*" script was added that manages all the objects that the player has looked at. It also saves the number of frames that user looked at these objects so that this can be later used to find the objects that user watched at the most (e.g. to analyse whether the looking-time affects later memory performance).

Also, a script was created and attached OVRPlayerController called "*LogSeen.cs*" that logs all the objects and the time when the user looked at them into a list of strings that can be later accessed during the experiment.

4 General Discussion

Virtual reality gives the researcher tools to manipulate and fully control the environments where experiments are conducted in. The technology is advancing in a fast pace and VR HMD's have become popular within the community of cognitive scientists [1,8,9,10]. It has been shown that virtual reality is a promising tool in the field of psychology and really immerses users into virtual reality [1].

VREX has been developed into a promising toolbox that has already been used in experiments by Vasser and others [15]. It gives the users a simpler more cost efficient way to create experiments in virtual reality. We expect VREX to be useful for researchers in the fields of memory, spatial navigation and attention, who want to use VR to conduct experiments, but do not wish to learn to use a whole game engine. VREX has all the necessary parts to build a typical experiment. We furthermore consider VREX to be a tool to replicate real life indoor environments in VR. Experimenters can manually place rooms, doorways and various objects. VREX can be used as a tool for interior design prototyping and to test different VR concepts, like many forms of locomotion.

During the development process we encountered many obstacles while trying to implement functionalities for VREX as a standalone application. We soon realised that going for a standalone application will restrict us as a developers and VREX itself. Every experiment will need custom functionalities and we cannot add them all. So going for a Unity project based solution seemed reasonable. This will give us more ways to implement functionalities and at the same time we will not need to worry whether they are working the same way in the editor and as a standalone application.

Keeping VREX a Unity project will also give users more freedom on making their own experiments. VREX should be used as a package of different pre built functionalities in addition to VREX giving the environment to conduct these experiments in.

We tried to add as many functionalities as we could come up with that we believed would be good basic functionalities. Now VREX needs feedback from users that would dictate how VREX will be developed further.

4.1 Future work

In our thesis we tested four different locomotion systems but tested only on 10 participants. In the future locomotion systems in VR should be looked into more closely because VR is getting popular and new VR applications being developed at a faster than ever pace. Right now many developers just find the solutions that they like but there has not been any scientific research on how to minimize the negative effects on HMD VR solutions.

Further development and bug fixing should be done to polish the functionalities that are already included in the project. These developments will need an actual feedback from other developers/users of VREX.

VREX will need definitely more new functionalities but they should all be suggested by users or even made by the users themselves. It would even be a good idea to create a forum especially for VREX if the user base gets big enough.

Known bugs:

- If user wants to build the application he must keep in mind that predetermined movement does not work. This bug was looked at during this thesis but a solution was not found.
- It has been noted that sometimes when new experiment is created even though the name is unique, VREX throws an pop-up window that says that there is already an experiment called like this.
- Teleportation script sometimes gives the error and says that player cannot teleport anywhere. This happens very rarely and no steps to reproduce this bug have been found.

Future work will be devoted to fix these bugs and improve the code.

4.2 Conclusion

In its current state VREX is ready for wider use. It has a series of functionalities that can be used to construct different psychological experiments and collect simple data into log files. More complex logging should be done based on the experiment.

Right now there are two example experiments and an empty experiment scene that can be used as a blank sheet for new experiments.

VREX requires a basic knowledge of Unity and comes with automatically generated documentation of all the scripts that are reasonably commented.

As a co-developer of this project I would recommend to distribute VREX as a open-source project toolbox and base future works on the feedback that can be received after VREX has been published and enough users have tried to create experiments using this toolbox.

REFERENCES

[1] A. Skulmowski, A. Bunge, K. Kaspar, G. Pipa, "Forced-choice decision-making in modified trolley dilemma situations: a virtual reality and eye tracking study", 2014. (2016, May) [Online]. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4267265/

[2] Oculus Rift kickstarter page. (2016, May) Kickstarter. [Online]. https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game

[3] S. Hayden. (2016, May) Road to VR. [Online]. http://www.roadtovr.com/oculus-reveals-175000-rift-development-kits-sold/

[4] Oculus Rift Development Kit 1 computer requirements page. (2016, April) Oculus Support. [Online]. https://support.oculus.com/hc/en-us/articles/201720623-Minimum-requirements-for-the-Oculus-Rift-Developer-Kit-1

[5] Oculus Rift Development Kit 2 computer requirements page. (2016, April) Oculus Support. [Online]. https://support.oculus.com/hc/en-us/articles/201835987-Oculus-Rift-Development-Kit-2-FAQ

[6] Oculus Rift retail version computer requirements. (2016, May) Oculus homepage. [Online]. https://www.oculus.com/en-us/blog/the-rifts-recommended-spec-pc-sdk-0-6-released-and-mobile-vr-jam-voting/

[7] HTC Vive retail version computer requirements. (2016, May) HTC Vive homepage. [Online]. https://www.htcvive.com/us/product-optimized/

[8] M. T. Schultheis, J. Himelstein, A. A. Rizzo, "Virtual Reality and Neuropsychology: Upgrading the Current Tools," Aspen Pub, 2002. (2016, May) [Online]. http://www.pages.drexel.edu/~sg94g745/Pubs/Virtual%20Reality%20and%20Neuroscience.pdf

[9] G. Hunter, Hoffman,J. Walter, Meyer, M. Ramirez, L. Roberts, E. J. Seibel, B. Atzori, S. R. Sharar,D. R. Patterson,*"Feasibility of Articulated Arm Mounted Oculus Rift Virtual Reality Goggles for Adjunctive Pain Control During Occupational Therapy in Pediatric Burn Patients*,", 2014. (2016, May) [Online]. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4043256/

[10] A. Felnhofer, O. Kothgassner, M.Schmidt, A .Heinzle, L. Beutl, H. Hlavacs, Kryspin-Exner, *"Is virtual reality emotionally arousing? Investigating five emotion inducing virtual park scenarios," International Journal Of Human-Computer Studies*, 2015. (2016, May) [Online]. http://www.sciencedirect.com/science/article/pii/S1071581915000981

[11] Oculus Rift simulator sickness, (2016, May), Oculus Rift homepage. [Online]. https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_app_simulator_sickness/

[12] Cloudhead Games - Blink VR Locomotion, (2016, May) Youtube. [Online]. https://www.youtube.com/watch?v=DwZt2jRE8PY

[13] OVR Player controller made by Oculus, (2016, May) Oculus Developer page. [Online]. https://developer.oculus.com/doc/0.1.2.0-unity/class_o_v_r_player_controller.html

[14] Runtime OBJ Importer by aaro4130, (2016, May) Unity Assetsstore. [Online]. https://www.assetstore.unity3d.com/en/#!/content/49547

[15] M. Vasser, M. Kängsepp, J. Aru, "Change Blindness in 3D Virtual Reality," 2015. (2016, May) bioRxiv database. [Online]. *http://dx.doi.org/10.1101/025817*

Appendix

Appenidx 1

A ".rar" format file called "*referencePoint.rar*"

Appendix 2

A ".rar" format file called "*VREX-Vladislav-Stafinjak.rar*"

Appendix 3

Our first player controller script.

https://drive.google.com/open?id=0B0i_w8ATfWOrc1V1OFlmaDI5RkE

Appendix 4

VREX adding new models video guide.

https://youtu.be/xhZ8n7seTxk

License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Vladislav Stafinjak**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, as of 12.05.2017 until expiry of the term of validity of the copyright,

Expanding the functionalities of VREX,

supervised by Jaan Aru, Madis Vasser and Raul Vicente Zafra,

2. I am aware of the fact that the author retains these rights.

3. This is to certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **12.05.2016**